Лабораторная работа №3

Выполнил студент группы Б9119-01.03.02миопд Гринёв Максим Владимирович

Интерполирование функции с помощью интерполяционных формул с конечными разностями

В этой лабораторной работе нам потребуется:

1. Построить таблицу конечных разностей для нашей функции.
2. Выбрав подходящие интерполяционные формулы, выполнить интерполирование табличной функции в точках $x^{**}, x^{***}, x^{****}$ используя максимально возможное количество узлов для каждой формулы.
3. Оценить погрешность интерполирования в этих точках, аналогично тому, как это сделано в лабораторной работе №1. Сравнить результаты с соответствующими значениями, получаемыми при непосредственном вычислении по формуле $y = f(x)$.

Задаем a, b, c:

```
import math
a, b = 0.4, 0.9
h = (b - a) / 10
```

Объявляем функцию:

```
def func(x):
    return x ** 2 + math.log(x)
```

Объявляем узлы и функции в узлах:

```
tableX = [int((a + i * h) * 100) / 100 for i in range(11)]
tableY = [func(tableX[i]) for i in range(11)]
```

Объявляем левые, правые, конечные разности:

```
tableLeft = [tableY[i + 1] - tableY[i] for i in range(10)]  # конечные разности
для левых
tableRight = [tableY[i] - tableY[i - 1] for i in range(10, 0, -1)]  # конечные
разности для правых
tableCenter = []  # конечные разности для центральных
```

Заполняем:

```
for i in range(int((len(tableY) - 1) / 2)):
    centerIndex = int((len(tableY) - 1) / 2)
    tableCenter.append(tableY[centerIndex] - tableY[centerIndex - i - 1])
    tableCenter.append(tableY[centerIndex + i + 1] - tableY[centerIndex])
```

Далее запишем в коде программы 1ую, 2ую формулу Ньютона, Гаусса и.т.д.:

```
def tNewton1(xStar):
    return (xStar - a) / h


def tNewton2(xStar):
    return (xStar - b) / h


def tGauss2(xStar):
    center = ((b - a) / 2) + a
    return (xStar - center) / h


def Omega(xStar):
    result = 1
    for i in range(11):
        if (tableX[i] < xStar):
            result *= (xStar - tableX[i])
    return result


def R(value, xStar):
    return func(value) * Omega(xStar) / 39916800.0


def Newton1(xStar):
    result = tableY[0]
    t = tNewton1(xStar)
    tRep = t
    n = 1
    for i in range(int(len(tableLeft))):
        n *= (i + 1)
        result += (tRep * tableLeft[i] / n)
        tRep *= (t - i)
    return result


def Newton2(xStar):
    result = tableY[10]
    t = tNewton2(xStar)
    tRep = t
    n = 1
    for i in range(int(len(tableRight)) - 1, -1, -1):
        n *= (len(tableRight) - i + 1)
        result += (tRep * tableRight[i] / n)
        tRep *= (t + (int(len(tableRight)) - i))
    return result
```

```
def Gauss2(xStar):  # t_0 + right_1 + t_0(2) + left_1 ...
    centerFunc = tableY[int(len(tableY) - 1 / 2)]
    result = centerFunc
    t = tGauss2(xStar)
    tRep = t
    n = 1
    sign = True
    j = 0
    for i in range(20):
        n *= (i + 1)
        if i % 2 == 0:
            result += tRep * centerFunc / n
        else:
            result += tRep * tableCenter[j] / n
            if sign:
                tRep *= (t + i + 1)
                sign = False
            else:
                tRep *= (t - i - 1)
                sing = True
            j += 1
    return result
```

Основная часть программы:

```
def Main(xStar, L):
    minR = R(b, xStar)
    maxR = R(a, xStar)
    # print(minR, "<", L - func(xStar), "<", maxR)
    print("True") if minR < L - func(xStar) < maxR else print("False")


print("1")
xStar2 = 0.43
Main(xStar2, Newton1(xStar2))
print("2")
xStar3 = 0.86
Main(xStar3, Newton2(xStar3))
print("3")
xStar4 = 0.67
Main(xStar4, Gauss2(xStar4))
```

Вывод программы:

```
1
True
2
True
3
True
```